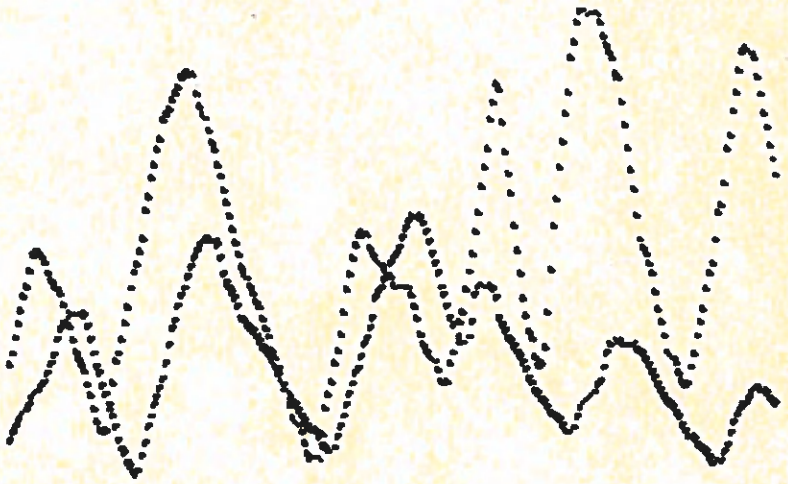
SINGLETON

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

VIDICHART™

By Paul K. Warme



```
BUFFER 00002 SCALE  00002
LEFT   00038 RIGHT  00293 SKIP   00001
BOTTOM 00002 TOP    00384 OFFSET-00001
CURSRX 00038 VALUE  00044 CURSRY 00002
```

Copyright (c) 1980

INTERACTIVE MICROWARE, INC.

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

V I D I C H A R T™

By Paul K. Warme

Copyright (c) 1980

INTERACTIVE MICROWARE, INC.

FEATURES OF VIDICHART 1
HOW TO PRODUCE A BACKUP DISK 2
CONTROL CHARACTERS FOR EDITING AND PROGRAM CONTROL 2
TUTORIAL DEMONSTRATION OF DISPLAY OPTIONS 3
    Practice with Scrolling 3
    A Position Report 4
    Changing the Active Buffer 4
    Changing the Scale 5
    Filling in a Curve 5
    Erasing a Curve 6
    Using the Cursors 6
ALPHABETICAL SUMMARY OF DISPLAY OPTIONS 8

HIGHER LEVEL FUNCTIONS OF VIDICHART 9

    DESCRIPTIONS OF INPUT/OUTPUT FUNCTIONS 9
PRT BUF$ SUB1# SUB2# >Print Buffer Values 9
INP BUF$ SUB1# VAL1# . . . VALN# >Input Buffer Values 9
LOD BUF$ SCALE# FILE$ >Load Buffer from Disk 9
SAV BUF$ SCALE# FILE$ >Save Buffer on Disk 10
ADC BUF$ CHANNEL# SAMPLE# DELAY# >A/D Converter Input 10
    EXPERIMENTING WITH INPUT/OUTPUT FUNCTIONS 11

    DESCRIPTIONS OF UTILITY FUNCTIONS 11
LEN BUF$ >Length of a Buffer 11
MAX BUF$ >Maximum Value in a Buffer 12
MIN BUF$ >Minimum Value in a Buffer 12
MOV BUF1$ SUB1# SUB2# BUF2$ SUB3# >Move Buffer Values 12
MVZ BUF1$ SUB1# SUB2# BUF2$ SUB3# >Move and Zero Rest 12
ZRO BUF$ SUB1# >Zero Part of a Buffer 12
POS BUF$ SUB1# >Position the Plot of a Buffer 13
    EXPERIMENTING WITH UTILITY FUNCTIONS 13

DESCRIPTIONS OF NUMERICAL FUNCTIONS 14
ADD BUF1$ BUF2$ BUF3$ >Add Buffer Values 14
SUB BUF1$ VAL# BUF2$ >Subtract Buffer Values 14
MUL BUF1$ BUF2$ BUF1$ >Multiply Buffer Values 14
DIV BUF1$ VAL# BUF1$ >Divide Buffer Values 14
DIF BUF1$ BUF2$ >Differentiate a Buffer 14
INT BUF1$ BUF2$ >Integrate a Buffer 14
AVG BUF1$ WIDTH# BUF2$ >Moving Average of a Buffer 15
EXPERIMENTING WITH NUMERICAL FUNCTIONS 15

DESCRIPTIONS OF SPECIAL FUNCTIONS 16
NRM BUF1$ BUF2$ >Normalize a Pair of Buffers 16
BAS BUF1$ BUF2$ >Define Baseline for a Buffer 17
PEK BUF1$ BUF2$ THRESH# WIDTH# CHANGE# >Integrate Peaks 18
EXPERIMENTING WITH SPECIAL FUNCTIONS 19

SUMMARY OF VIDICHART ERROR MESSAGES 20

USER MODIFICATIONS OF VIDICHART 21
Adding Functions 21
Useful BASIC Subroutines 22
USR Argument Summary 22
How to Use a Different A/D Converter 23

SAMPLE PICTURES

UPDATES
Update #1
Update #2

VIDICHART CHANGES FOR USE WITH SILENTYPE PRINTER

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

NOTES ON PROGRAM LOADING

All Interactive Microware programs are supplied either in DOS 3.2 format or in a special "double boot" format which can be booted on either a DOS 3.2 (13-sector) or DOS 3.3 (16-sector) disk system. The label on the disk indicates whether the disk is DOS 3.2 or DOS 3.2/3.3 (double boot) format. If you are using the DOS 3.3 system, the programs can be converted from the DOS 3.2 format, as supplied, to a DOS 3.3 format disk by using the MUFFIN program (on your System Master disk). For your convenience, the programs are not copy protected; however, please note that the programs are copyrighted, so it is illegal to make copies, except for your own use.

DOS 3.2 USERS:

You should be able to boot, load, or copy programs from any of our disks labelled DOS 3.2 or DOS 3.2/3.3.

DOS 3.3 USERS:

If the disk is labelled DOS 3.2, you must boot your system with the BASICS diskette or mount the System Master disk and type BRUN BOOT13. Then, when the prompting message on the screen tells you to mount the DOS 3.2 (13-sector) disk, insert our program disk and press RETURN.

If the disk is labelled DOS 3.2/3.3, turn off the computer and insert our program disk and turn the computer on. The DOS 3.2 operating system is first loaded from our disk, and then the HELLO program is run on a DOS 3.2 system.

To convert our programs to DOS 3.3 form, boot the DOS 3.3 System Master disk, type BRUN MUFFIN, and follow the instructions to copy the desired programs to a disk that has been INITialized in DOS 3.3 format.

FEATURES OF VIDICHART

Vidichart makes it easy for experimenters to input, display, compare and perform numerical operations on large sets of data. High-speed machine language subroutines control the high resolution display of up to four data sets of 1024 points, using a distinct plotting symbol for each. Under keyboard control, any of the four curves may be scrolled (non-destructively) to the left or right, up or down and the curve may be expanded or contracted in the X or Y direction. Two different cursors are provided: the tracking cursor follows along a plotted curve, whereas the free cursor can move anywhere on the screen under keyboard control. At any time, a printed report of the display status may be superimposed on the bottom four lines of the screen. This report gives complete information about the position of the left, right, top and bottom limits of the screen, scaling information, the position of the cursor and its value.

But Vidichart is much more than just a fancy plotting program; it includes higher-level functions to facilitate input and output of data and to perform numerical operations on entire sets of data. Data may be entered from the keyboard, from a disk file or from an analog to digital converter. Output of data may be directed to the screen, to a printer or to the disk. The straightforward format of the data stored on disk makes it easy to use data produced by other programs, such as the Scientific Plotter and Curve Fitter programs by this same author or many of your other programs. Utility functions of Vidichart permit you to determine the maximum value, minimum value or length of any data set, move selected data from one buffer to another or zero out some or all of the data values in a given data set. Numerical operations include addition, subtraction, multiplication or division of a data set by another data set or by a constant, differentiation, integration and N-point smoothing (i.e., moving average).

Vidichart also includes some special functions which are useful for chromatography, spectrophotometry and other disciplines. A normalization function adjusts one data set to have the same average value as another data set. The baseline generator function allows you to define a baseline by using the free cursor to enter the end points of line segments. The peak integrator function automatically lists all of the maxima and minima for a curve and integrates each distinct peak. Several adjustable parameters may be used to fine tune the peak-finding algorithm.

This manual explains how to use each of the many features of Vidichart and includes a series of demonstrations which will give you hands-on experience with this powerful and versatile set of data management tools.

HOW TO PRODUCE A BACKUP DISK

Although this program is copyrighted, you have permission to make copies for your own use on a single machine. Thus, your first action should be to copy the master disk to another disk as follows:

1. This program is designed to run on a 48K Apple II computer with Applesoft in ROM. Mount the master disk, type RUN VIDICHART and then press RETURN (from now on, you should always press RETURN after entering a command.) After a short time, the screen will be erased, the copyright notice will appear and then the mini menu prompt "1=IN/OUT 2=UTILITY 3=NUMERIC 4=SPECIAL" with a flashing cursor at the end of the prompt.

2. Type Control Q (hold down the CONTROL key and press the Q key) to stop the program.

3. Mount your slave disk and type SAVE VIDICHART.

4. Save the machine language part of the program by typing BSAVE VISIOBJ,A$8A00,L$7F0. The program actually uses all of the space from $8A00 to $95FF, but the highest 1024 bytes are used for data.

CONTROL CHARACTERS FOR EDITING AND PROGRAM CONTROL

Control H (back arrow) deletes the last character that you typed. Note that the right arrow key is ignored by this program while in text mode.

Control X prints a back slash, advances to the next line and ignores the entire input line.

Control G switches to Graphics mode so that you can examine or modify plots of your data, as described below.

Control T causes Vidichart to return to Text mode so that you can enter a high-level function command.

Control P allows you to select a Print device for subsequent output. After the Control P, type a single digit number denoting the slot occupied by the printer controller or type 0 to restore normal screen output.

Control D is the signal that the preceding command typed on the input line is a Disk Operating System (DOS) command. For example, typing CATALOG followed by Control D would list the disk catalog. You might also use this feature to delete or rename files, etc. A different drive, slot or volume may be selected by appending the new number to the CATALOG command. For example, CATALOG, D3, V10, S5 selects drive 3, volume 10 and slot five. Subsequent disk accesses will continue to use that disk.

Control Q means to Quit the program and return to BASIC. VIDICHART may be restarted at line 10 (GOTO 10) without destroying your variables.

All other Control keys are ignored during user input in text mode; however, Control S may be used to pause during screen output and Control C will stop the program (unless it is waiting for user input; see Control Q).

TUTORIAL DEMONSTRATION OF DISPLAY OPTIONS

Let's jump right into the heart of Vidichart with this hands-on demonstration. Type RUN VIDICHART (or just RUN if it is already loaded). First, we will load some sample data from the disk file called DATA1 into buffer A. Mount the master disk and type CTRL U to turn off the prompts, then type

```
LOD A 1 DATA1
```

Next, we'll load a different set of data (DATA2) into buffer B. Type:

```
LOD B 1 DATA2
```

The LOD command will be explained later; for now, let's just try the display. Type Control G to enter the Graphics mode. Hmmm . . . the screen just blanks out. Now, type 1 (or Control A) to select buffer A and then press P to Plot buffer A. Two hundred and fifty six symbols immediately appear on the screen. If you press P again, the symbols will disapppear. That is due to the way the display routines work; each time a point is drawn, it reverses the color of the screen at that position. This approach prevents one curve from erasing another as it moves around on the screen. Anyway, knowing how this works will help you to understand some other peculiar effects that may occur later under unusual circumstances. Press the P key again and hold it down while pressing the REPT key. This makes the display flash. You may find this flashing effect useful later, when the screen is crowded and you want to highlight one of the curves.

Practice with Scrolling

Next, press the left arrow key and note that the display shifts each symbol to the left and a new symbol appears on the right. If you press the left arrow key and the REPT key at the same time, the points continuously scroll to the left. Likewise, the right arrow key shifts each point to the right and the REPT key may be used for continuous scrolling. Use the semi-colon key (;) to shift the display down & the colon key (:) to shift the display up. Try the REPT key. Isn't that nice? Four-way scrolling of a graphics display!

If you scroll buffer 1 (A) too far to the right, you will run out of points to display and then, you will see "garbage" on the left side of the screen. Likewise, if you scroll too far to the right, you will run out of points from buffer 1 and the rightmost points will actually be taken from buffer 2 (B). Bear in mind that the four buffers, 1 through 4, are packed right next to one another in memory. The display system does not check to make sure that you are paying attention to buffer boundaries. No harm is done when you exceed the limits of a buffer, as long as you understand what is happening. The Report command will help you keep track of where you are.

A Position Report

After moving the display around for awhile, you may forget where you started out. Press the R key if you want a Report of the display status. The bottom four lines tell you that BUFFER number 1 (A) is active and the SCALE factor is 1. The number of the point displayed at the left edge of the screen is given after LEFT and the point on the right edge is shown after RIGHT. The value given after SKIP should be 1. This is the interval between the points being displayed; for example, if SKIP is 2, every second point is displayed. BOTTOM tells you the Y axis value of a hypothetical point plotted at the very bottom of the screen, while TOP gives the equivalent Y value for a hypothetical point plotted at the top of the screen. The OFFSET value tells you the number of steps the curve has been displaced above or below the original baseline. For example, if OFFSET is 10, the colon (up) key has been pressed 10 times; if OFFSET is -10, the semi-colon (down) key has been pressed 10 times. Let's ignore CURSRX, VALUE and CURSRY for now. Just for practice, why don't you try to reposition the curve at LEFT=0 and OFFSET=0. You will notice that the report disappears whenever you press any key, but you can quickly recall it by pressing R.

Changing the Active BUffer

Next, let's see what buffer 2 (or B) looks like. Press 2 (or Control B) and then P to Plot buffer 2. You will notice that the scrolling keys now work on buffer 2, instead of buffer 1 and the Report (try typing R) reflects the values for buffer 2. Only one buffer can be active at a given time, but you can promptly switch from one buffer to another by pressing any of the number keys from 1 through 4 or the corresponding Control keys from A to D. Buffers 3 and 4 are transparent now, because we haven't put any data in them. As the curves scroll past one another, they do not erase each other unless two points momentarily coincide.

You will notice that the plotting symbols used for buffer 2 are different from those used for buffer 1. There is a pattern here. Symbol 1 looks like an L; you may think of it as the hands on a clock showing 12:15. Symbol 2 is rotated clockwise 90 degrees and looks like 6:15 on the clock. Symbol 3 is rotated clockwise again and looks like 6:45 and symbol 4 looks like

11:45. When symbols are plotted close together, overlap can cause parts of the symbols to disappear. This is due to the screen color inversion technique used to plot the points; plotting the first symbol turns the points on, but an overlapping symbol may turn those points off again.

Changing the Scale

There is something peculiar about buffer 2. The heads are shaved off some of the peaks. Vidichart has another set of commands that allows you to change the scale for either the X axis or the Y axis. Press H and the height of the active curve is Halved. You can press H repeatedly and even use the REPT key, but the curve gets very small after a few times. To increase the height of a curve, press the D key to Double its size. When the curve reaches its original size, the D key no longer has any effect. The C and E keys perform a similar function, but they affect the X axis instead of the Y axis. Pressing C Contracts the curve, whereas the E key Expands the curve along the X axis. Again, you may press these keys repeatedly or use the REPT key, but eventually you will reach a limit, beyond which the curves cannot be compressed or expanded any further. With one exception (see the F command below), only 256 points are shown on the screen for each buffer at any given time.

If you press the R (Report) key, you can observe the effect of the D, H, E and C keys on the reported values. Note that each time you press C (and then R), the value of SKIP increases by 1; in other words, the interval between the plotted points increases. Conversely, the E key decreases SKIP. If SKIP is larger than 4, the points on the right side of the screen are from the next buffer; this is apparent from the fact that RIGHT is greater than 1023. In fact, at the maximum compression (SKIP=16), all four buffers can be viewed at the same time. If OFFSET>0, when you type H, the value of BOTTOM decreases and the value of TOP increases; in other words, the values attached to points plotted at the bottom and top of the screen are becoming larger in absolute magnitude. Conversely, pressing D will increase BOTTOM and decrease TOP. However, if the zero line is exactly at the bottom of the screen (OFFSET=0), the H and D keys always leave BOTTOM equal to zero.

Filling in a Curve

By this time, you may have noticed that when you Compress a curve, the spacing between adjacent points appears to increase. This is because the SKIP interval increases and some of the points are not plotted. You can Fill in the missing points by pressing the F key. This causes every point in the active buffer to be plotted. But replotting the points that were already on the screen erases them, so you should press P tomake them reappear. It is not possible to scroll more than 256 points, though, so if you move the curve or change scale, only

256 points will move and the rest of the filled-in points will remain where they were (try this). Thus, before moving the curve, you may want to erase the other points by pressing F again (and also press P if necessary to make the original 256 points reappear).

Erasing a Curve

You already know that you can erase the active curve by pressing P. However, if you move the curve after erasing it this way, the curve will reappear double. This is because the previous curve (at its old position) is redrawn in the process of "erasing" it, and then the curve is drawn at its new position. This effect can be useful if you really want to have two copies of the same curve on the screen at the same time.

A different way to erase the active curve is to press the X key. This zeroes out all memory of where the points were plotted previously, so pressing X again will not make the curve reappear. You will have to press P to replot the curve.

If things get out of hand and you want to start over with a fresh slate, you can use the Y key to erase the entire screen. This works the same as repeated use of the X key on every buffer. Now, you can replot each of the curves, one at a time, by typing 1P, 2P, etc.

Using the Cursors

Vidichart provides two different types of cursors. The "tracking" cursor follows from point to point along the active curve and is useful for determining the point number or the Y value of a particular point. The "free" cursor can move anywhere on the screen and is used to enter position information in both X and Y directions. For example, the free cursor is used for defining a baseline. To obtain information about the cursor position, press the R (Report) key. The bottom line now shows CURSRX, the point number in the active buffer where the cursor is resting. Also shown is the VALUE of Y for that point. CURSRY is the same as VALUE for the tracking cursor, but if you are using the free cursor, CURSRY will tell you the equivalent Y value for that position on the screen.

Now, let's see how to use the tracking cursor. When you press K (for Kursor), a small plus (+) sign appears at one position on the screen, right on top of one of the points of the active buffer. It may be hard to see the cursor if it is located at the bottom of the screen or in a congested zone, so try pressing the K and REPT keys at the same time. This causes the cursor to wink. Next, press the J key. This makes the cursor move to the left, whereas pressing L makes it move to the right. You can remember these keys by noting that they are on the left and right side of the K key. Also, they are close to the REPT key, so you can scan rapidly by pressing either key at the same

time as the REPT key. Try using REPT. When the cursor reaches the right edge of the screen, it jumps back to the left edge and vice versa. You should also try the R key and see how the values of CURSRX, VALUE and CURSRY change as you move the cursor.

The free cursor appears when you press the O key. The circle of the O should call to mind that the free cursor can move in any direction. Pressing O and REPT simultaneously makes the free cursor wink. The J and L and REPT keys may be used to move the free cursor to the left and right, but it doesn't track along the active curve, does it? You can also move the free cursor up by pressing I and down by pressing M. Note that these keys are the same as the ESC keys used to move the cursor in Applesoft and that they form a logical cluster at right angles to the K key. Use the R key to get a Report of the cursor position and observe how CURSRX, VALUE and CURSRY change as you move the free cursor around on the screen. When you are through experimenting, type Control T to return to Text mode.

ALPHABETICAL SUMMARY OF DISPLAY OPTIONS

CTRL A or 1	Activate buffer A (1)
CTRL B or 2	Activate buffer B (2)
CTRL C or 3	Activate buffer C (3)
CTRL D or 4	Activate buffer D (4)
CTRL G	Switch to Graphics mode
CTRL T	Switch to Text mode
<-	Scroll active curve left
->	Scroll right
:	Scroll up
;	Scroll down
REPT	Repeat any key operation
C	Compress X scale
D	Double size on Y axis
E	Expand X scale
H	Halve size on Y axis
I	Move free cursor up
J	Move cursor left
K	Select tracking cursor
L	Move cursor right
M	Move free cursor down
O	Select free cursor
P	Plot active buffer
X	Erase active curve
Y	Erase entire screen

HIGHER LEVEL FUNCTIONS OF VIDICHART

The following functions are available whenever you are in Text mode. If you are currently in Graphics mode, type Control T to enter Text mode. Each function is requested by a three letter command, followed by a space and one or more arguments separated by spaces. The type of argument is indicated by a $ sign for strings or a # sign for numbers. BUF$ denotes a buffer letter from A to D, SUB# means a subscript between 0 and 1023 and other numbers, such as VAL#, must be integers between -32767 and 32767, except as noted.

DESCRIPTIONS OF INPUT/OUTPUT FUNCTIONS

PRT BUF$ SUB1# SUB2# >Print Buffer Values
e.g., PRT A 0 10

The PRT function PRinTs the values contained in buffer BUF$ at subscript positions SUB1# through SUB2#. The present subscript value is printed at the beginning of each line of five numbers. You may use the Control P command to obtain a hard copy.

INP BUF$ SUB1# VAL1# . . . VALN# >Input Buffer Values
e.g., INP A 0 10 15 20

The sequence of numbers VAL1# through VALN# will be stored in buffer BUF$, starting at position SUB1#. Up to six lines of numbers may be entered, with a single space after each number.

LOD BUF$ SCALE# FILE$ >Load Buffer from Disk
e.g., LOD A 2.5 DATA1

This will LOaD successive values from disk file FILE$ into buffer BUF$, starting at subscript position zero. Each value will be multiplied by SCALE# (a floating point value is OK) before storing it. The purpose of this scale factor is to allow you to convert floating point numbers in a disk file to integers for use in this program, without significant loss of accuracy. For example, if your disk file contains numbers between -10.014 and 20.650, you might use a scale factor of 1000 to convert them to a range of -10014 to 20650. As the numbers are stored, they are integerized, so any fractional part is lost.

The format of disk files used with VIDICHART is very simple and is identical to that used by Scientific Plotter and Curve Fitter. The first item in the file is the number of values stored in this file. Then come the values themselves. For example, if D(N) is an array of N values, the following program

would create file F$, which is compatible with VIDICHART:

```
100 PRINT CHR$(4)"OPEN "F$: PRINT CHR$(4)"WRITE "F$
110 PRINT N: FOR I = 0 TO N-1: PRINT D(I): NEXT
120 PRINT CHR$(4)"CLOSE "F$
```

If a disk file contains more than 1024 numbers, only the first 1024 will be read. A missing file name will cause "FILE NAME ERROR" to be printed. If any other error occurs during a LOD operation, VIDICHART prints "ERROR CODE", followed by a number which refers to the error codes listed in the Applesoft manual (p. 81) or the DOS 3.2 manual (pp. 114-115).

SAV BUF$ SCALE# FILE$ >Save Buffer on Disk
e.g., SAV A 1. DATA2

The SAV command will SAVe all of the data from buffer BUF$ on the disk as file FILE$. Before each number is transferred, it will be multiplied by SCALE#, which may be a floating point number. If SCALE# is one, the buffer values will be saved in their original form. If you want to switch disk drives before saving a file, you should type CATALOG, followed by the new drive (e.g., D2), slot and/or volume number and then type Control D.

Here is a program that you could use to read a disk file (F$) created by VIDICHART into one of your own programs:

```
200 PRINT CHR$(4)"OPEN "F$: PRINT CHR$(4)"READ "F$
210 INPUT N: FOR I = 0 TO N-1: INPUT D(I): NEXT
220 PRINT CHR$(4)"CLOSE "F$
```

The error messages for SAV are the same as those for LOD.

ADC BUF$ CHANNEL# SAMPLE# DELAY# >Analog to Digital Converter Input

This function reads a number of samples (SAMPLE#) from a given Analog to Digital Converter channel (CHANNEL#) using a delay factor of DELAY#. The values are placed sequentially in BUF$, starting at position 0.

Since you could use any A to D converter for this function, VIDICHART allows you to customize the routine to suit your hardware, as described in the section entitled User Modifications of VIDICHART. However, in the meantime, you can test the ADC function using the APPLE game controls and the standard version of VIDICHART. In this version, CHANNEL# must be either 0 or 1 to select one of the two game controls and DELAY# must be between 0 and 32767 or else a RANGE ERROR will result. Set DELAY# to 50 for the first try or use a lower number for faster sampling. After you enter the ADC command, turn the dial on the selected game control and you will see points plotted on the screen. If you

have requested more than 256 points, the plotter will jump from the right edge back to the left edge and erase the previous points as the new points are plotted. After the sampling is finished, VIDICHART replots the buffer with position 0 on the left side of the screen and returns to text mode. To view your handiwork, type Control G.

EXPERIMENTING WITH INPUT/OUTPUT FUNCTIONS

Just reading about these features gets pretty dry, so let's give these commands a test run. (If you have already tried them, that's great! Charge on and read about the Utility Functions.) If you haven't already fired up VIDICHART, type RUN VIDICHART.

First, let's see what is currently stored in buffer C. Type PRT C 0 10. All zeroes. Well, let's put something more interesting in there. Type INP C 0 0 10 20 30 40 50 60 70 80 90 100. This places the numbers from 0 to 100 in buffer C, starting at position 0. To confirm that it worked, type PRT C 0 10. You might also want to check the plotted results (type Control G 3 P and then Control T to return to Text mode).

Next, let's save this data on the disk. Mount your slave disk and type SAV C 1 TEST1. Since we're using a scale factor of 1 here, the data will be stored without change. In order to make sure that file TEST1 was actually saved, type CATALOG and then Control D. TEST1 should be listed as a text file of 2 blocks. Now, how about loading this data into buffer D? Before loading the data, type PRT D 0 10 to verify that buffer D is empty at this time. Then type LOD D 2 TEST1. How will the scale factor of 2 used here affect the values in buffer D? Type PRT D 0 10 to see if you're right.

It's time to have some fun with the ADC command now. I'll assume that you're using the standard version of VIDICHART, which uses the game controls for input. Type ADC A 0 500 50 to read 500 values from game control 0 with a delay of 50 and place the reults in buffer A. You should have your hand on the game control knob before pressing RETURN, because things happen quite rapidly once the action starts. Twiddle the knob to make the curve rise and fall. After the program returns to text mode, type Control G to view the graph and scroll it around, compress or halve it. Type Control T when you are through playing around. You can repeat the ADC command as many times as you want to and experiment with changing the buffer, the channel number, the number of samples and the delay factor. With a delay of 0, you can sample at a rate of more than 50 samples per second.

DESCRIPTIONS OF UTILITY FUNCTIONS

LEN BUF$ >Length of a Buffer

e.g., LEN A

This prints the message, LENGTH = and then a number which tells the position (subscript) number of the last non-zero value. If values 1 through 1023 are all zero, LENGTH is reported as 0, regardless of whether the value at position 0 is zero or nonzero.

MAX BUF$ >Maximum Value in a Buffer
e.g., MAX B

Prints the message, MAXIMUM = and then the number which is the largest of all numbers in the specified buffer.

MIN BUF$ > Minimum Value in a Buffer
e.g., MIN C

Prints MINIMUM =, followed by the smallest of all numbers in the buffer.

MOV BUF1$ SUB1# SUB2# BUF2$ SUB3# >Move Buffer Values
e.g., MOV A 0 100 B 10

The MOV command MOVes the values from subscript positions SUB1# through SUB2# of buffer BUF1$ to sequential positions in BUF2$, starting at subscript SUB3#. If BUF2$ is the same as BUF1$ and SUB3# is less than SUB2#, the results will be incorrect, due to overlap. In this case, you must first move the data from BUF1$ to a different buffer, BUF3$ (MOV BUF1$ SUB1# SUB2# BUF3$ SUB1#) and then move the data from BUF3$ back into BUF1$ (MOV BUF3$ SUB1# SUB2# BUF1$ SUB3#).

MVZ BUF1$ SUB1# SUB2# BUF2$ SUB3# >Move and Zero Remainder
e.g., MVZ B 100 200 A 0

MVZ performs just like MOV, except that after the transfer is completed, the remaining values (that is, those beyond the last value stored) in BUF2$ are set to zero. This is useful when you are moving values into a buffer that has already been used for something else and you want to clean up the garbage.

ZRO BUF$ SUB1# >Zero Part of a Buffer
e.g., ZRO C 200

This stores zeroes in all values from SUB1# onward in buffer BUF$. If SUB1# is less than zero, its value is negated (made positive) and then all values preceding that position in the buffer are deleted as the remaining values are moved down to start at subscript 0. After this shift is completed, the unused part of the buffer is zeroed. For example, if buffer A contains

Ø 1 2 3 4 5 Ø Ø . . . and you issue the command ZRO A -3, buffer A will now contain 3 4 5 Ø Ø . . ..

POS BUF$ SUB# >Position the Plot of a Buffer
e.g., POS A 1Ø

You may find it inconvenient sometimes to scroll a buffer to a particular position on the display screen. The POS command enables you to replot the display with a given data point (SUB#) at the left edge of the screen. Since this is a text mode command, you must type Control T to exit from graphics mode, type the POS command and then type Control G to return to graphics mode. BUF$ becomes the active buffer after the POS command.

EXPERIMENTING WITH UTILITY FUNCTIONS

Feel free to skip this section if the utility commands are completely clear in your mind. If you haven't already done so, type RUN VIDICHART and use the INP, LOD or ADC commands to enter some values in buffer A.

Let's see how many values are stored in buffer A. Type LEN A. What is the maximum value? Type MAX A. And what is the minimum value in A? Type MIN A. You will notice that LEN returns a value almost immediately, but MAX and MIN take longer; their time is proportional to the length of A. This is also true of most of the other text mode functions. If you are working with large buffers of data, you'll have to be patient; BASIC is working its little heart out. While you're waiting, think about how long it would take you to do all of those calculations by hand.

Next, we'll move some values from buffer A to buffer B. Type PRT A Ø 1Ø and then PRT B 1Ø 2Ø to see what is currently stored in these buffers. Now, type MOV A Ø 1Ø B 1Ø. Check the results by typing PRT B 1Ø 2Ø. As an example using MVZ, let's move the same values from buffer A to position Ø in buffer B. Type MVZ A Ø 1Ø B Ø and then PRT B Ø 2Ø to verify that values 11 and onward are now zero. Another way to verify this is to use the LEN B command, which should return LENGTH=1Ø.

We can continue using buffer B to demonstrate the ZRO function. First, type ZRO B 6 and then PRT B Ø 1Ø. All of the values beyond position 5 should be zero now. If you type ZRO B -5 followed by PRT B Ø 1Ø you will see that the value formerly at position 5 is now at position Ø and all of the other values are zero.

The final demonstration concerns use of the POS command. Type POS A Ø and then Control G to enter Graphics mode. Next, type R for a Report and verify that LEFT is zero. Then, type Control T to return to Text mode. Where would you like to

position the display? Type POS A #, where # stands for any position number that pleases you. Go back to Graphics mode, observe the plotted curve and verify that the value of LEFT is the same as you requested.

DESCRIPTIONS OF NUMERICAL FUNCTIONS

ADD BUF1$ BUF2$ BUF3$ >Add Buffer Values

SUB BUF1$ VAL# BUF2$ >Subtract Buffer Values

MUL BUF1$ BUF2$ BUF1$ >Multiply Buffer Values

DIV BUF1$ VAL# BUF1$ >Divide Buffer Values
e.g., DIV A 10.1 B

These commands are presented as a group because their format and arguments are identical. The selected operation is performed on each value of BUF1$, the first argument. The second argument can be either another buffer (BUF2$) or a floating point constant (VAL#). If it is a buffer, the values stored at corresponding positions in BUF1$ and BUF2$ are used for the operation. The results are placed in corresponding positions in the buffer given as the third argument. Note that the third argument may be the same as the first argument. If any result is less than -32767, the value -32767 is stored and if any result is greater than 32767, it is stored as 32767. Also, any fractional part of the result is truncated to an integer.

DIF BUF1$ BUF2$ >Differentiate a Buffer
e.g., DIF A B

This command DIFferentiates (in other words, takes the derivative of) a buffer. After this operation, the first value in BUF2$ is the same as the first value of BUF1$, the second value of BUF2$ is the difference between the second and first values of BUF1$, the third value of BUF2$ is the difference between the third and second values of BUF1$, and so on. You may specify the same buffer for both BUF1$ and BUF2$.

INT BUF1$ BUF2$ >Integrate a Buffer
e.g., INT A B

INT stands for INTegrate, which is just the opposite of DIF. The operation is performed by taking a running total of the values in BUF1$ and the current total is placed in the corresponding position of BUF2$. If the sum is less than -32767 or greater than 32767, the closest integer value is stored in buffer BUF2$, although the running total is carried accurately as a floating point number. It is permissible to select the same

buffer for both BUF1$ and BUF2$.

Notice that DIF and INT are reciprocal operations; that is, one undoes the other. For example, DIF A A followed by INT A A leaves A exactly as it was originally and vice versa.

AVG BUF1$ WIDTH# BUF2$ >Moving Average of a Buffer
e.g., AVG A 10 B

The AVG function computes the moving AVeraGe of WIDTH# successive values in BUF1$ and stores the results in corresponding positions of BUF2$. Another name for this function is N-point smoothing, because it tends to smooth out bumps in a curve. Usually, the purpose of averaging is to smooth out random noise in a set of observations.

If BUF2$ is the same as BUF1$ or WIDTH# is greater than the length of BUF1$, the ARGUMENT ERROR message will be printed.

EXPERIMENTING WITH NUMERICAL FUNCTIONS

If you haven't already done so, type RUN VIDICHART to crank this program up. In order to simplify these examples and start with known data, let's first zero out buffers A and B (type ZRO A 0 and then ZRO B 0). Then, type INP A 0 0 1 2 3 4 5.

To exercise the ADD function, let's add 1 to each value in buffer A and put the results in buffer B (type ADD A 1 B). You should use the PRT function after each step of our procedure in order to verify that the results are correct. We can also add a buffer to itself and place the result in the same buffer (type ADD A A A). There are no restrictions on the identities of the buffers used in the ADD, SUB, MUL and DIV functions. However, only the second buffer can be replaced by a constant.

Here is a little puzzle for you: using only the SUB command only twice, how can you make buffer A and B identical (but not all zeroes)? From this point, you should be able to transform buffer B so that it contains 0 0 4 4 8 in two steps, using the MUL command once and the DIV command once. The solution is in the next paragraph, so don't peek ahead unless you give up.

This is just a spacer to keep you from reading the answer prematurely. Just in case you're a fast reader and your eyes jump ahead here, it's not too late to turn back and try the puzzle. It's too late now; subtract 1 from B and leave the result in B. Then subtract B from A, leaving the result in A. Divide A (or B) by 2 and multiply the result by 4, leaving the result in B. Notice the effects of integer truncation during division.

For practice with the DIF, INT and AVG functions, it helps to

have a fair-sized buffer with several peaks. Mount the VIDICHART master disk and type LOD A 1 DATA2. Then, type DIF A B. To view the results, type POS A Ø and POS B Ø and then type Control G. Since some of the values in buffer B are negative, scroll curve B upward. You will notice that the derivative curve (B) has twice as many peaks as curve A. Also, the derivative curve changes sign at each peak or valley in curve A. Now, let's integrate buffer B and see whether the result is identical to A. Type Control T, INT B B and then Control G. Curve B will be updated if you erase it (type 2X) and then redraw it (type P). If curve B was scrolled upward previously, scroll it downward until it is superimposed on curve A. If they are identical, both curves should disappear when they are exactly aligned. However, due to the fact that different plotting symbols are used for the two buffers, some fragments of the symbols will remain on the screen.

As our final exercise, we'll carry out ten point averaging on buffer A and place the result in B. Type Control T to get back in Text mode, then type AVG A 1Ø B. Type Control G and then 2XP to update curve B. You will notice that curve B is now smoother than curve A and the peaks and valleys are less sharp. When comparing complex curves like this, it helps to press P and REPT simultaneously, so that one curve blinks.

DESCRIPTIONS OF SPECIAL FUNCTIONS

NRM BUF1$ BUF2$ >Normalize a Pair of Buffers
e.g., NRM A B

The purpose of normalization is to correct for differences in scale when you are comparing two sets of data. This is most useful when the two buffers are quite similar in their pattern of peaks and valleys. The most straightforward way to normalize two curves is to take the ratio of the values of two corresponding maxima in the pair of curves and then multiply the lower of the two curves by this constant ratio. However, this approach suffers from the limitation that it relies on the accuracy of a single point from each curve. Moreover, on sharp peaks, it may be impossible to find a point which is exactly at the peak. Therefore, VIDICHART uses an improved method of normalization. Each of the two buffers is integrated over the same range of points and buffer BUF1$ is then multiplied by the ratio of the two integrals, leaving the results in BUF1$. If the result is outside the range -32767 to 32767, the greatest integer value is stored in BUF1$. Since BUF1$ is changed by the NRM operation, you should MOVe a copy of BUF1$ to a different buffer if you want to save the original data. Also, note that it is usually desireable to choose the higher curve as BUF1$, since this will minimize integer truncation errors.

Before normalization, it is important to ensure that both curves share the same baseline (zero point) and that the two

curves are aligned properly; that is, corresponding points in the two buffers should fall at the same positions in each of the two buffers. The BAS and SUB functions may be used to adjust the baseline and the ZRO function may be used to trim excess points off the left end of the longer of the two buffers. Excess points on the right end of one of the buffers do no harm, because the length of the shortest buffer is taken as the range of integration. All points in BUF1$ are multiplied by the ratio of the two integrals, even when BUF1$ is longer than BUF2$.

BAS BUF1$ BUF2$ >Define Baseline for a Buffer
e.g., BAS A B

In many cases, the baseline (or zero point) of a curve is not flat across the entire curve. When possible, it is a good idea to evaluate the baseline exactly, store the baseline values in a second buffer and then use the SUB command to subtract the baseline curve from the data curve. However, sometimes it is not possible to evaluate the baseline directly, so it is necessary for the experimenter to estimate the baseline, based on the data curve itself. The BAS function is designed to help you with this task. BUF1$ is the buffer containing the uncorrected data and BUF2$ will hold the baseline data after completion of this procedure. If the results are satisfactory, you should then use the SUB command to subtract BUF2$ from BUF1$.

After you enter the BAS command, the program enters Graphics mode, BUF1$ is replotted with position 0 on the left edge of the screen and BUF2$ is zeroed. The program automatically invokes the free cursor, which should appear on the left side of the screen. (Type O and REPT together to make the cursor blink). Use the I and M keys to move the free cursor up or down to define the starting position of the baseline. Then, type Control T to register that position. Now, use the I, J, L and M keys to move the cursor to the next endpoint of a straight-line segment of the baseline. Again type Control T to register that position. VIDICHART will interpolate a straight line between these points, store the results in BUF2$ and plot the line. Continue using the I, J, L, M and Control T keys to enter additional line segments until you approach the right side of the screen. If the curve is wider than 256 points, you will want to shift both the main curve and the baseline curve to the left now. VIDICHART assumes that this is what you want to do whenever you enter a cursor position that is to the left of the previous cursor position. In this event, both curves will be replotted so that the position of the cursor is at the left edge of the screen. In other words, if you want to shift both curves to the left, move the cursor leftward to the point that you want to occur at the left edge of the screen and press Control T. The BAS procedure ends when you enter a cursor position beyond (to the right of) the end of the main curve.

One precaution should be noted when using the BAS function.

The values of SCALE, OFFSET and SKIP should be the same for both curves (BUF1$ and BUF2$), otherwise the baseline will not be plotted correctly. Use the R key to check this if you're not sure about these values or the baseline isn't plotted correctly. All of the graphic control keys work the same as usual while the BAS procedure is going on, so you are allowed to change the position, the X scale or the Y scale of the main curve (BUF1$). Just remember to make the same changes to the baseline curve (BUF2$).

PEK BUF1$ BUF2$ THRESH# WIDTH# CHANGE# >Find and Integrate Peaks
e.g., PEK A B 10 20 30

In chromatography, spectrophotometry and other disciplines, it is useful to integrate individual peaks within a curve. The PEK function searches for peaks in buffer BUF1$ and prints a report showing the positions of minima and maxima and the integral of each peak. To obtain a hard copy of the printed report, type Control P and a single digit slot number before typing the PEK command. A MINimum in the printout is defined as the first point that rises above the threshold. If the integral of the previous peak has not been printed yet, the integrated value is printed and a new integral is started with this point. A MAXimum in the printout is the point having the largest value within the current peak. A CUT means that the curve has gone below the threshold, so integration of the previous peak terminates and the integrated value is printed. The TOTAL of all integrated peaks is printed at the end of the listing.

The cumulative integrals for each peak are stored in BUF2$ so that you can visually determine the limits of integration for each peak. To make a visual comparison, type Control G and align BUF1$ and BUF2$ with equivalent corresponding points at the left edge of the screen. Reduce the Y scale of BUF2$ by typing H a few times. You should see a series of S shaped curves for BUF2$ and the end points of the S's should coincide with valleys in BUF1$. If you don't see a separate S curve for each peak in BUF1$ or the endpoints of the S's appear to come at the wrong positions, you may be able to improve the results by fine-tuning the integration variables, as discussed below.

The integration variables THRESH#, WIDTH# and CHANGE# are optional; one, two or all of them may be omitted. The default value is 5 for each of them. THRESH# is the THRESHold for starting and ending the integration of a peak. It is assumed that values below this threshold represent the baseline or random noise. WIDTH# is the minimum width (number of points) recognized as a peak. The purpose of this variable is to filter out small noise spikes that occur at values above THRESH# and are superimposed on a larger peak. If the number of points between the current minimum or maximum and the previous valid minimum or maximum is less than WIDTH#, the current peak will be ignored. CHANGE# is the minimum change in height that must occur between a

peak and a valley in a valid peak. Its purpose is to decrease the sensitivity of the peak finding algorithm so that it doesn't find small peaks that barely get above the baseline and avoids false peaks due to noise excursions.

EXPERIMENTING WITH SPECIAL FUNCTIONS

If you haven't already fired up VIDICHART, type RUN VIDICHART. Mount the master disk and type LOD A 1 DATA1.

For practice with the NRM command, let's create a new data set that is double the size of DATA1 and then normalize them. Type MUL A 2 B. You can check the results by using PRT to print out some values or else by shifting into graphics mode and plotting the two curves. Next, type NRM A B. Again check the values in buffers A and B. The values in A and B should now be identical and buffer A should be twice as large as it was originally. Try changing some of the values in A or B and again normalize them. You will notice that the results are quite insensitive to small changes in one of the buffers, especially if the sum of the positive changes is about the same as the sum of the negative changes.

We can use the same data to try out the BAS command. Type BAS A B. Press O and REPT at the same time so the cursor on the left side of the screen flashes. Use the I or M key to move the cursor near to the bottom of the screen and type Control T. Now, move the cursor about 3 inches to the right (press L and REPT) and up or down a bit (I or M) and type Control T. A line will be drawn between the two endpoints. Repeat this procedure until the baseline has grown to about an inch from the right end of the curve. Now, back the cursor up about an inch (J) and type Control T. Both curves will shift to the left so that the last inch from the right of the previous display is now on the left side of the screen. Now move the cursor to the right, past the last endpoint on the baseline and add more line segments. When the last line segment extends beyond the last point on the main curve, the BAS procedure quits and the program returns to text mode. Next, type SUB A B C to subtract the baseline and place the results in buffer C. Switch to graphics mode and compare buffers A, B and C. You might want to use the POS command to line them up at corresponding positions.

Our last experiment concerns the PEK function. We might as well use the curve in buffer C, which has a corrected baseline. This is the usual sequence of operations that you would follow with your own experimental data. Type PEK C D. After awhile, a list of the MINima, MAXima and CUToff points will appear on the screen, together with the INTEGRAL for each peak. Now, shift to Graphics mode and compare the integral curve (4) to the sample curve (3). Do the S's in curve 4 begin and end at the valleys in curve 3? Feel free to experiment with different values of the THRESH#, WIDTH# and CHANGE# variables. How much change is required to significantly affect the integrated results?

SUMMARY OF VIDICHART ERROR MESSAGES

If VIDICHART encounters a mistake in your command line, it will print one of these error messages, followed by a copy of your command with a question mark inserted at the location of the error.

COMMAND ERROR: The first three letters of your command are not a recognized command.

MISSING BUFFER: A letter from A to D was not found at the position where a buffer name is required.

MISSING NUMBER: A numeric character was not found at the position where a number is required. Only the numbers from Ø through 9, a decimal point, a minus sign or the letter E before an exponent are recognized as numbers.

RANGE ERROR: A number outside the allowable range was detected. Subscripts must be between Ø and 1Ø23 and other values must be between -32767 and 32767.

FILE NAME ERROR: The file name is missing in a LOD or SAV command.

ERROR CODE: An error occurred during a LOD or SAV operation. The number following this message refers to one of the codes listed in the Applesoft manual (p. 81) or the DOS 3.2 manual (pp. 114-115).

ARGUMENT ERROR: In an AVG command, either the number of points in the buffer is less than the number of points to be averaged or you have not selected two different buffers.

REQUIRES 2 BUFFERS: The PEK command requires two different buffers.

USER MODIFICATIONS OF VIDICHART

Adding Functions

No doubt, some of you have some ideas for improvement of VIDICHART. These notes should help you to avoid some pitfalls. It is very important that you don't change anything below line 54 in VIDICHART, because this will mess up the report that appears on the bottom four lines when you type R in graphics mode. The memory space for these lines falls right in the middle of the program. (If you list VIDICHART, you'll see some related garbage in lines 38, 45, 49 and 53).

Lines 90 to 99 are reserved for defining variables. All variables used by VIDICHART are listed here. Unless you are sure you understand what you're doing, avoid using these variables in any statements you add to the program.

It is fairly easy to add additional functions to VIDICHART. Let's assume that you want to add a function called NEG that NEGates each value of a buffer. (Of course, you could also do this by zeroing a buffer and then subtracting your buffer from the buffer containing zeroes.) Let's say you want to be able to put the result in a second buffer. Thus, a typical command would be NEG A B. All statement numbers above 1010 are available for your program, so we'll put it at 2000. First, you must change line 800 to read

```
800 IF CM$ < > "PEK" GOTO 2000
```

and add line 2000

```
2000 IF CM$ < > "NEG" GOTO 42
```

Note that all command names are 3 letters long and the first argument must be a buffer in all cases. After you type a command, the function name is returned in CM$, the first buffer number is in B0 and the length of that buffer is returned as N. The next task is to get the second argument from the command line. In this case, we're expecting a buffer name again, so we will call subroutine 46, which returns the next buffer as B0 and the length of the buffer in N. But we must first save B0 and N for the first buffer, so include the line:

```
2010 B1 = B0: N1 = N: GOSUB 46
```

All of the data for buffers is stored in an array called D%(1023,3), so buffer A has a second subscript of 0, B is 1 and so on. The remainder of the NEG program is easy:

```
2020 FOR N = 0 TO N1: D%(N,B0) = -D%(N,B1): NEXT
```

We don't care about the length of the second buffer, so we can use variable N as a counter. In fact, once you have extracted all arguments from the command line, you can use any of the variables listed in line 91. On completion of any function, jump to line 10 to input the next command.

```
2030 GOTO 10
```

Useful BASIC Subroutines

Here are some subroutines that you may find useful:

Subroutine 46 returns the name of the next buffer in B0 and its length in N. If the next argument isn't a buffer, it prints MISSING BUFFER and quits (jumps to 10).

Subroutine 50 does the same as 46, but doesn't quit if the next argument is not a buffer. Instead, it returns RG=-1 if it is a buffer and RG=0 if not.

Subroutine 54 looks for a numeric argument as the next item of a command. If a number is found, its value is returned as N and RG is set to 1. Otherwise, RG will be 0 on return.

Subroutine 66 does the same as 54, except that it prints MISSING NUMBER and quits if the next argument is not a number.

Subroutine 70 prints RANGE ERROR and quits if N is less than 0 or greater than 1023, the valid range for subscripts.

Subroutine 74 prints RANGE ERROR and quits if N is less than -32767 or greater than 32767, the valid range of integers.

Subroutine 80 may be used to switch to graphics mode but remain in BASIC.

USR Argument Summary

Most of the machine language functions of VIDICHART are invoked by passing the Ascii value of the corresponding letter as the argument of the USR function. For example, U=USR(ASC("E")) or U=USR(69) will Expand the active buffer. This same approach will work for all other display options except those outlined here. Control A (Ascii 1) through Control D (Ascii 4) must be used to select the active buffer; the numbers 1 through 4 (Ascii 49 to 52) are not recognized. USR(5) switches to graphics mode, not USR(7) as you might expect for Control G. After USR(5), the machine language program remains in control until Control T is typed. The P command is handled differently when called from BASIC. To replot the active buffer, the argument of USR must be 256 plus the number of the point that you want to appear on the LEFT edge of the screen. To zero a buffer in the forward direction, the USR argument is 2048 plus the number of the first point to be zeroed. To zero a buffer in the reverse direction, the USR argument is -2048 minus the number of the first point to be kept. The length of a buffer is returned as U when you include the statement U=USR(B+9), where B is a buffer number between 0 and 3. U=USR(0) initializes the display system, zeroes all buffers and clears the screen.

The easiest way to learn how to use these USR commands is to study the examples in the BASIC portion of VIDICHART. Have fun, but please bear in mind that this program is copyrighted and should not be distributed to others in any form.

How to Use a Different Analog to Digital Converter

This section explains how to adapt VIDICHART to use an A/D converter other than the game controls. Consult the instructions that came with your A/D board to determine the correct procedures for selecting the channel number, starting a conversion and retrieving a value. Usually, this will involve POKEing the channel number at a particular location, POKEing another location to start a conversion and then, after a suitable delay, PEEKing the sampled value at one location for an 8 bit converter or two locations for more than 8 bits. In the latter case, you will have to multiply the high order byte by 256 and add the low order byte.

Now, let's look at subroutine 1000, which VIDICHART calls when you enter an ADC command. (List lines 1000 to 1010). Information from your ADC command is passed to subroutine 1000 in the following form: B0 is the buffer to be used, OP is the channel number, NS is the number of samples to be taken and DLY is the delay between measurements. GOSUB 80 switches to graphics mode and then U = USR(B0+1) + USR(-3071) activates B0 and zeroes the buffer. The ADC routine uses a special display option, the N (Ascii 78) command, which plots the Next point referenced by a counter stored at location XC (35365 decimal). This counter value is automatically updated by the N display option, so all we have to do is set the XC counter to 0 at the outset. Line 1005 checks the value of OP and prints RANGE ERROR if OP is not a valid channel number. If your A/D converter requires a channel number, insert the code to select the channel here.

The sampling is actually performed in line 1010. NS samples are read from the A/D converter and stored in the D% array (D%(I,B0) = PDL(OP)). Typically, you should replace this statement with a POKE to start the conversion and a PEEK to read the value and store it in D%(I,B0). The U = USR(N) statement plots the current point on the screen, and erases the point plotted previously at that position on the screen. Next comes a delay loop (FOR J = 0 TO DLY: NEXT) to control the rate of sampling. If you are using a slow A/D converter, it may be necessary to insert this delay loop between the start conversion command and the read sample value command. After completion of the sampling loop, the buffer is replotted starting at position 0 (U = USR(H)) and the subroutine returns after switching to TEXT mode. This approach will support a sampling rate of up to 50 samples per second.

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

VIDICHART UPDATE #1

1. A "cueing" feature has been added to VIDICHART in order to remind you of the command formats. First, you select one of the four categories of functions (Input/Output, Utility, Numerical or Special Functions) by typing a number from 1 to 4. VIDICHART then prints the three-letter names of all functions in that category, and you may select the desired function by typing the number preceding its name. Next, the function name is printed at the beginning of the line, followed by a list of the arguments that you must enter for that function. If the argument name begins with a $ sign, your response should be a letter or group of letters. An argument name beginning with a # sign should be answered with a number. You may enter all of the arguments at once, with a space between each argument and a RETURN at the end of the line. Alternatively, you may enter one argument at a time and press RETURN, in which case VIDICHART will prompt you if additional arguments are required. The correctness of your responses is not checked until all arguments have been entered, so you may backspace (left arrow) to correct any errors that you detect before completing the list of arguments. To delete the entire command, type Control X. As usual, you may switch to graphics mode by typing Control G.

2. After you become more experienced in using VIDICHART, you may wish to turn off the "cueing" prompts so that you can work faster. To do this, type Control U (for User cue). Later, if you get stuck and can't remember the syntax of a command, just type Control U again to return to "cueing" mode. In other words, Control is like a toggle switch that turns the "cueing" mode on or off.

3. In the graphics mode, the R (Report) key has been changed to a toggle switch. Thus, after you type R, the report remains on the screen until you type R again or you type Control T to return to BASIC. It is quite helpful to watch the numbers change as you scroll or change the scale.

4. During the BAS function, the R (Report) switch is automatically turned on each time you enter a new point on the baseline, so that you can read the cursor position. You will usually want to turn the report off again by pressing R before you move the cursor to the next point on the baseline.

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

VIDICHART UPDATE #2

1. A new "SAV" feature has been added which allows you to save and load data in binary files, rather than in text files normally used by VIDICHART. This speeds up disk accesses by more than a factor of 2. To select binary disk input or output, use #SCALE=Ø in the SAV or LOD command (e.g., SAV A Ø DATA or LOD B Ø BINARY). The binary SAV and LOD routines always save or load all 1Ø24 points of a buffer, so you won't have to zero out the data previously stored there. The binary data on the disk is stored in high byte before low byte order, as usual for a BASIC array. Using the binary SAV routine, you can also save hi-res screen memory page 2 (your graph) by choosing a file name starting with the letters PIC. This is handy if you want to print the picture using a graphics printer utility program.

2. Only if you are using the Silentype version of VIDICHART, make the following change, which causes the program to terminate on Control S (not Control Q) when in the "cueing" mode.

```
993 IF C=19 THEN END
```

3. In the "cueing" mode, you may abort a command at any time by typing Control X.

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

VIDICHART CHANGES FOR USE WITH SILENTYPE PRINTER

If you own a Silentype printer, the changes described below will enable you to print your graphs at the touch of a key. Also, you will be able to print the series of questions and answers that produced the graph, tables of results and other information that you wish to save.

LOAD **VIDICHART** and type the following changes exactly:

```
32  IF C < 22 GOTO 974

974  IF C = 21 GOTO 970
975  IF C = 19 THEN  END
976  IF C = 17 THEN  POKE  - 12529,255: POKE  - 12528,7: POKE  - 12525,64:
      POKE  - 12524,0: PRINT  CHR$ (17)
977  IF C = 20 THEN  PRINT  CHR$ (C): POKE  - 12529,0
978  IF C <  > 18 GOTO 14
979  FOR C = 2640 TO 3024 STEP 128: FOR A = C TO C + 38: PRINT  CHR$ ( PEEK
     (A) - 128);: NEXT : PRINT : NEXT : GOTO 10
```

Now, save this modified version on your own disk (do not remove the write protect tab from the original disk). To use the printer, you must turn off the "cueing" prompts by typing CTRL U. Then, type CTRL P, followed by a single digit number that tells what slot your Silentype interface card is in. For example, CTRL P1 would select slot 1. From this point on, each line that you type should appear both on the screen and on the printer. If the screen display (echo) is off, type CTRL T; each time you type CTRL T, the screen echo feature is toggled on or off. The PRT command does not TAB correctly unless the screen echo is on.

After the printer is activated (by CTRL P), you may print your graph by merely typing CTRL Q. However, CTRL Q is ignored by the display subsystem, so if you are in Graphics mode, type CTRL T to return to Text mode (in Graphics mode, CTRL T does not toggle the screen echo). You may also print the on-screen Report by typing CTRL R while in Text mode. To Stop **VIDICHART**, type CTRL S instead of the usual CTRL Q.

Interactive Microware, Inc.
P.O. Box 771
State College, Pa 16801
(814) 238-8294

CHANGES IN SCIENTIFIC PLOTTER, CURVE FITTER AND VIDICHART

FOR USE WITH EPSON MX-7Ø AND EPSON MX-8Ø PRINTERS AND FOR THE PKASO* PRINTER INTERFACE SERIES

The changes below assume that you are using either the EP-12 parallel interface card (made by Interactive Structures, Inc.) or the GRAPPLER(tm) interface card (made by Orange Micro, Inc.). Both of these cards feature ROM routines that enable you to print the high resolution screen by typing just a few control characters. If you do not already own one of these printer interface cards, we highly recommend the new EP-12 card because it has special commands to print low resolution graphics, half-tone graphics, super high resolution graphics (96Ø x 792 points) and special characters and symbols defined by you. The EP-12 also works with Pascal and CP/M systems, and it includes a DEMO DISK showing examples of all features. The price of this powerful interface card is only $175, and it is now available from Interactive Microware.

For both Scientific Plotter and Curve Fitter, make the following changes for use with the EP-12 interface:

```
2852 IF C=16 THEN GET C$: PRINT: PRINT CD$+"PR#"+C$:
     PRINT CHR$(9)"8ØP": GOTO 281Ø

2858 IF C=17 THEN PRINT CHR$(9)"15H": GOTO 281Ø

2859 IF C=18 THEN TEXT: END
```

For the GRAPPLER interface, omit the PRINT CHR$(9)"8ØP": in line 2852 and change "15H" in line 2858 to "G2". After making these changes, you can turn on the printer at any time by typing CTRL P, followed by a single digit number denoting the slot that your interface card is plugged into. If the printer is on, you can print the hi-res screen by merely typing CTRL Q. To stop the program, type CTRL R instead of CTRL Q (as in the manual).

* For the PKASO series of interface cards (by Interactive Structures, Inc.), change "80P" to "I" and change "15H" to "33H".

The modifications for VIDICHART closely follow the changes described in the manual for use with the Silentype printer. Thus, make the same changes in lines 32, 974, 975, 978 and 979, and alter line 976 and 977 as follows:

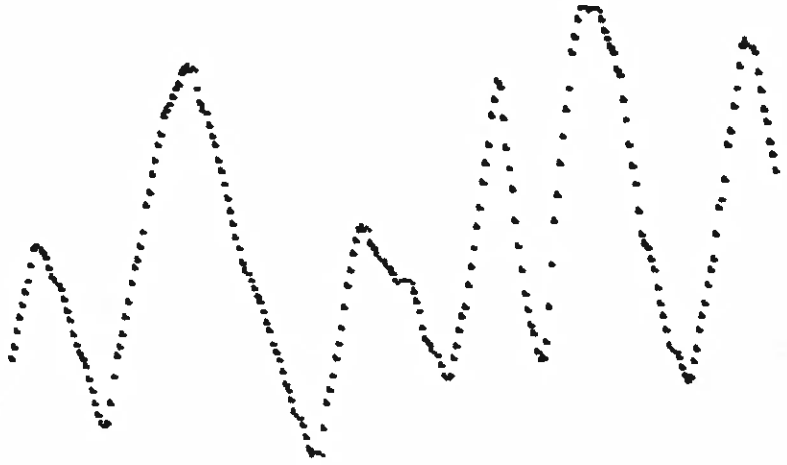
976 IF C=17 THEN PRINT CHR$(9)"15H" (or "G2" for the GRAPPLER).

977 IF C=2Ø THEN PRINT CHR$(9)"8ØP"

Bear in mind that you must turn off the "cueing" mode (type CTRL U) before activating the printer by typing CTRL P, followed by the slot number. When the printer is on, type CTRL Q to print the screen image and type CTRL R to print the report at the bottom of the screen. To stop the program, type CTRL S. In text mode, typing CTRL T when the printer is on will turn on the screen echo so that characters are printed on both the screen and the printer.

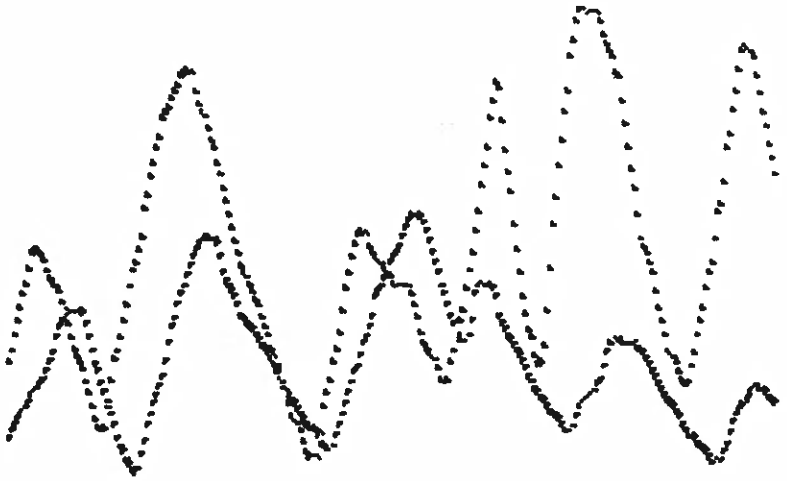
VIDICHART EXAMPLES

DATA1



```
BUFFER 00001 SCALE  00001
LEFT   00000 RIGHT  00255 SKIP   00001
BOTTOM 00000 TOP    00191 OFFSET 00000
CURSRX 00000 VALUE  00036 CURSRY 00000
```

DATA1 + DATA2



```
BUFFER 00002 SCALE  00002
LEFT   00038 RIGHT  00293 SKIP   00001
BOTTOM 00002 TOP    00384 OFFSET-00001
CURSRX 00038 VALUE  00044 CURSRY 00002
```

APPLE COMPUTER, INC. MAKES NO WARRANTIES,
EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED
COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY
OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.